Iranian Journal of Operations Research
Vol. 13, No. 1, 2022, pp. 48-60

Applying Branch and Cut Method to a Graph Based Reduction of *UCTP*

M. Khorramizadeh[۱*]

Here, we first associate a graph to a university course timetabling problem (UCTP) and use the components of this graph and some customary and organizational rules to transform the original large scale problem into some smaller problems. Then, we apply the branch and cut method to obtain the optimal solution of each smaller problem. Our presented approach enables us to apply exact methods to obtain high quality solutions for large scale UCTPs. Finally, we examine the numerical efficiency of the resulting algorithm.

***Keywords**: Integer Programming, University Course Timetabling, Branch and Cut Method, Binary Variables, Scheduling Problem.*

Manuscript was received on 04/15/2022, revised on 10/03/2022 and accepted for publication on 01/23/2023.

۱. Introduction

The timetabling problem is common to academic institutions such as schools, colleges or universities. Educational timetable generation is one of the major administrative requirements in scientific institutions. It is a very hard combinatorial optimization problem which attracts the interest of many researchers. University course timetabling problem falls in the category of *NP*-hard problems having various constraints, objectives, and limited resources. Generating an optimized timetable is challenging and time-consuming process.

The university course timetabling problem (*UCTP*) is to allocate time periods and rooms to courses, such that the rules of the educational organization are satisfied. Two important types of this problem are the curriculum-based and the enrollment-based university course timetabling. Courses which can be taken, because they are needed to satisfy the degree rules of the given study, are called curricula. An allocation of rooms and time periods to courses based on this, is called curriculum-based course timetabling. In the enrollment-based case, courses are placed in the timetable such that all students can attend the lectures on which they are enrolled [17, 18]. Here, we focus on the curriculum-based university course timetabling. The curriculum based university course timetabling problem consists in determining the best scheduling of university course lessons in a given time interval, assigning the lessons of each course to classrooms and time periods, so that a series of constraints is satisfied. These constraints are divided into two categories: hard constraints, necessary so that the programming can actually be implemented, and soft constraints, which involve qualitative measures [11].

The *UCTP* is a combinatorial problem. For the first time it was presented in [13]. The *UCTP* is *NP*-hard problem [12], but it has a great practical relevance, for instance [15, 20]. In [3], an iterated

[*] Corresponding Author.
[۱]Assistant professor, Shiraz University of Technology, Iran, Email: m.khorrami@sutech.ac.ir.

local search algorithm is proposed to find the feasible solution for the *UCTP*. Three key phases are involved in the proposed algorithm framework: initialization, intensification and diversification. Once a partial-feasible initial timetable is constructed, a simulated annealing based local search and a diversification procedure that brings moderate perturbation or even improvement to the current solution are performed in an iterative manner until a stop condition is met. Colajanni and Daniele [11] deals with the study of the curriculum based university course timetabling problem. This paper formulates a new and complete model that satisfies both the planning constraints and those on the compactness of the curricula, the distribution of the lessons (in the examined time frame), the teachers' preferences, the minimum number of working days, maximum capacity and stability of the classrooms (which aims to minimize the daily movements of students among classrooms) so that the resulting timetable is of high quality. The formulated model, with appropriate adaptations, has been applied to the real case study of the first year of the Mathematics Degree Course of the University of Catania, Italy.

Since university timetabling is commonly classified as a combinatorial optimization problem, researchers tend to use optimization approaches to reach the optimal timetable solution. Meta-heuristic algorithms have been presented as effective solutions as proven on their leverage over the last decade. However, a comprehensive systematic overview is missing. Evan et al. [12] aimed to provide an organized view of the current state of the field and comprehensive awareness of the meta-heuristic approaches, by conducting meta-heuristic for solving university timetabling problems. In addition, the mapping study tried to highlight the intensity of publications over the last years, spotting the current trends and directions in the field of solving university timetabling problems, as well as having the work to provide guidance for future research by indicating the gaps. Komijan and Koupaei [15] describes the decision support system that was developed for the assignment of courses to teaching modalities and rooms for the Fall semester of 2020 at the University of Connecticut (*UConn*). With the adoption of safety/mitigation standards imposed by the *COVID-19* pandemic, the seating capacities of rooms were reduced by more than 70%, thus making virtually every existing room assignment for Fall 2020 infeasible. In order to maximize opportunities for in-person instruction, UConn introduced a teaching modality in which class meetings are attended on campus by only 50% of the enrolled students. As decision makers were given partial flexibility to assign teaching modalities to classes, the complexity of the assignment problem increased considerably, especially because the real-world instances involved hundreds of rooms and thousands of classes and required a quick solution turnaround in practice. In this article, they introduce this flexible assignment problem and describe the two mixed-integer programming formulations that were used to solve the real-world instances of the problem. The examination timetabling problem can be described as a set of exams to be scheduled over an examination session while respecting numerous hard and soft constraints. Chen et al. [9] considers the spacing soft constraints that seek to prevent students sitting more than one exam per day. Kaur et al. [14] presents a study for a local search algorithm based on chromatic classes for the university course timetabling problem. Several models and approaches to resolving the problem are discussed. The main idea of the approach is through a heuristic algorithm to specify the chromatic classes of a graph in which the events of the timetable correspond to the graph vertices and the set of the edges represents the possible conflicts between events. Then the chromatic classes should be sorted according to specific sort criteria (a total weight or a total count of events in each class), and finally the local search algorithm starts. The aim of the experiments is to determine the best criterion to sort chromatic classes. The results showed that the algorithm generates better solutions when the chromatic classes are sorted in a total weight criterion.

Overviews [2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 19, 20] examine the university course timetabling problem, focusing on mathematical models, one and multi-objective, exact and heuristic algorithms in the literature.

Here, we present a graph based reduction approach for the university courses timetabling problem. We first associate a graph to the *UCTP*. Each node of this graph corresponds to a teacher of the university. If two teachers, teach some courses of the same group, then we define an edge between the corresponding nodes. Next, we use the components of this graph along with some customary and organizational rules to transform the original large scale problem into smaller ones and use the branch and cut method to solve each of them. Finally, we present some numerical results to examine the efficiency of our proposed approach.

In section 2 we describe the characteristics of *UCTP*. In section 3 we study the integer programming formulation of *UCTP*. Section 4 is concerned with our proposed graph based reduction approach and finally, in section 5 we examine the numerical results.

۲. Integer Programming Formulation

In this section, we describe the terminology, notations and the integer programming formulation of UCTP. Let $C=\{1,2,...,\bar{c}\}$, $R=\{1,2,...,\bar{r}\}$, $T=\{1,2,...,\bar{t}\}$, $D=\{1,2,...,\bar{d}\}$, $G=\{1,2,...,\bar{d}\}$ and $S=\{1,2,...,\bar{s}\}$ be the set of courses, rooms, time periods, days, groups and teachers, respectively. For any $s \in S$, let $C_s \subseteq C$ be the set of courses taught by teacher s. For any $g \in G$, let $C_g \subseteq C$ be the set of courses attended by group g. The course c should be scheduled in n_c time periods per week. If the course c is scheduled for more than one days a week, then the number of time periods that are scheduled for that course in day d should be between $n^c_{\min}$ and $n^c_{\max}$. Moreover, we let τ_d be the first time slot of day $d \in \mathrm{D}$. For simplicity in notation we assume that we do not consider any break between morning and evening sessions. This way, all time periods of the first day are in $[\tau_1, \tau_2)$ and so on.

Next, we define some binary variables. If the course $c \in C$ is scheduled at time t and in room r, then x_{crt} is one, otherwise x_{crt} is zero. If in day $d \in \mathrm{D}$ the course $c \in C$ is taught, then, u_{cd} is one, otherwise u_{cd} is zero. If in day $d \in \mathrm{D}$ the teacher $s \in S$ teaches at least one course, then, ψ_{cd} is one, otherwise ψ_{cd} is zero. The university course timetabling problem must satisfy the following constraints.

• For any course $c \in C$, n_c time periods in a week must be scheduled.

$$\sum_{r \in R} \sum_{t \in T} x_{crt} = n_c, \quad c \in \mathrm{C}, \tag{1}$$

• At time $t \in T$, any class $g \in G$ can attend at most one course.

$$\sum_{c \in C_g} \sum_{r \in R} x_{crt} \leq 1, \qquad g \in G,\ t \in T, \tag{2}$$

• At time $t \in T$, any teacher s can teach at most one course.

$$\sum_{c \in C_s} \sum_{r \in R} x_{crt} \leq 1, \quad s \in S, t \in T, \tag{3}$$

• At time $t \in T$, the room $r \in R$ can host at most one course.

$$\sum_{c \in C} x_{crt} \leq 1, \quad r \in R, t \in T, \tag{4}$$

• If course $c \in C$ is taught in day $d \in \mathrm{D}$, then the teaching hours of course $c \in C$ must be between $n^c_{\min}$ and $n^c_{\max}$.

$$\sum_{r \in R} \sum_{\tau_d \leq t \leq \tau_{d+1}} x_{crt} \leq n^c_{\max} u_{cd}, \quad \mathrm{c} \in \mathrm{C}, d \in \mathrm{D}, \tag{5}$$

$$\sum_{r \in R} \sum_{\tau_d \leq t \leq \tau_{d+1}} x_{crt} \geq n^c_{\min} u_{cd}, \quad \mathrm{c} \in \mathrm{C}, d \in \mathrm{D}, \tag{6}$$

• If two time periods are scheduled for the same course in day $d \in \mathrm{D}$, then they have to be assigned to adjacent time periods.

$$\sum_{r \in R} \left(x_{crt_1} - x_{crt_2} + x_{crt_3} \right) \leq 1, \quad \mathrm{c} \in \mathrm{C}, d \in \mathrm{D},\ \tau_d \leq t_1 < t_2 < t_3 \leq \tau_{d+1}, \tag{7}$$

• At most one room can be assigned to the time periods of each course in every day.

$$x_{cr_1t_1} + x_{cr_2t_2} \leq 1, \qquad c \in C, d \in D, \tau_d \leq t_1 < t_2 \leq \tau_{d+1}, 1 \leq r_1 < r_2 \leq \bar{r}, \tag{8}$$

• If a room or a teacher is unavailable we set the corresponding variable $x_{crt} = 0$.

Next, we study the objective function of the problem. Clearly, the objective function is the minimization of the measure of the sum of undesirabilities. To define the measure of the sum of undesirabilities we consider two approaches. In the first approach we let p_{crt} be the measure of the undesirability of the assignment of course $c \in C$ to room r in time period t. In this approach, the objective function is the minimization of the sum of penalties i.e.

$$\min \sum_{c \in C} \sum_{r \in R} \sum_{t \in T} p_{crt} x_{crt} \tag{9}$$

In the second approach, each desirability is represented by a constraint in such a way that if the constraint is satisfied, then the corresponding desirability is satisfied. These constraints are called soft constraints. The amount of violation of a soft constraint, reflects the amount of the corresponding undesirability. Therefore, we try to minimize the measure of the sum of undesirabilities, by minimizing the sum of violation of these constraints. In our presented approach, we consider the following soft constraints.

• For each group $g \in G$, empty time periods between any two courses are not allowed.

$$\sum_{c\in C_g}\sum_{r\in R}\left(x_{crt_1} - x_{crt_2} + x_{crt_3}\right) \le 1, \quad g \in G, d \in D, \tau_d \le t_1 < t_2 < t_3 \le \tau_{d+1}, \tag{10}$$

• Each class attend at most $l_{\max}$ teaching time periods in a day.

$$\sum_{c\in C_g}\sum_{r\in R}\sum_{\tau_d \le t \le \tau_{d+1}} x_{crt} \le l_{\max}, \qquad g \in G,\ d \in D, \tag{11}$$

• A teacher $s \in S$ works at most k_s days a week.

$$\sum_{s\in S} \psi_{sd} \le k_s, \quad d \in D, \tag{12}$$

It is clear that all kind of desirabilities, cannot be represented by using constraints. For example, assume that a teacher would not like his courses to be scheduled in rooms, far from his office. Clearly, it is difficult to find a constraint to represent this desirability. On way to deal with this difficulty is to use a combination of the first and second approaches to define the objective function. The resulting objective function is as follows.

$$\min \left[\sum_{g\in G}\sum_{d\in D}\left(\left\{ \sum_{d \le t_1 < t_2 < t_3 \le \tau_{d+1}} \sum_{c\in C_g}\sum_{r\in R}\left(x_{crt_1} - x_{crt_2} + x_{crt_3} - 1\right)\right\} + \left\{ \sum_{c\in C_s}\sum_{r\in R}\sum_{\tau_d \le t \le \tau_{d+1}} \left(x_{crt} - l_{\max}\right)\right\}\right) + \left(\sum_{s\in S}\sum_{d\in D}\left(\psi_{sd} - k_s\right)\right) + \sum_{c\in C}\sum_{r\in R}\sum_{t\in T} p_{crt}x_{crt} \right] \tag{13}$$

Next, we list some valid inequalities relating to the integer programming formulation of the *UCTP* described in the previous section. These valid inequalities are used to improve the efficiency of the branch and cut method and are introduced by Avella and Vasilev [2]. The validity and effectiveness of these inequalities can be found in [2].

• Min (Max) Busy Days inequalities:

$$\sum_{d\in D} u_{cd} \le \left\lfloor \frac{n_c}{n^c_{\min}} \right\rfloor, \quad c \in C \tag{14}$$

$$\sum_{d\in D} u_{cd} \geq \left\lfloor \frac{n_c}{n^c_{\max}} \right\rfloor, \quad c \in C \tag{15}$$

• Adjacency inequalities:

$$-x_{crt-1} + x_{crt} - x_{crt+1} \leq 0, \quad t \in \left[\tau_{d+1}, \tau_{d+1} - 2\right] \tag{16}$$

$$x_{crt} - x_{crt+1} \leq 0, \quad t = \tau_d, \ d \in D \tag{17}$$

$$-x_{crt-1} + x_{crt} \leq 0, \quad t = \tau_{d+1} - 1, \ d \in D \tag{18}$$

Where, $C = \left\{c \in C : n^c_{\min} \geq 2\right\}$.

• Inequalities

$$\sum_{r\in R}\sum_{t\in T_{cdt_1}} x_{crt} + \sum_{r\in R}\sum_{t\in T_{cdt_2}} x_{crt} = u_{cd}, \quad c \in C, d \in D, \ \tau_d \leq t_1 < \tau_d + n^c_{\max}, \ \iota_d \leq t_2 < \iota_d + n^c_{\max} \tag{19}$$

$$C = \left\{c \in C : n^c_{\min} = n^c_{\max}\right\}$$

Where, $T_{cdt_1} = \left\{t = t_1 + k n^c_{\max} : k \in \mathbb{N} \cup \{0\}, \ \tau_d \leq t < \iota_d\right\}$

$$T_{cdt_2} = \left\{t = t_1 + k n^c_{\max} : k \in \mathbb{N} \cup \{0\}, \ \iota_d \leq t < \tau_{d+1}\right\}$$

• Inequalities

$$x_{cr_1t_1} + x_{cr_2t_2} - \bar{u}_{cd} \leq 1, \quad c \in C, 1 \leq r_1 < r_2 \leq \bar{r}, \ d \in D, \ \tau_d \leq t_1 < t_2 < \tau_{d+1} \tag{20}$$

Where, $\bar{u}_{cd} = 1 - u_{cd}$, $c \in C, d \in D$.

۳. Graph based reduction of *UCTP*

In this section we describe our presented approach for graph based reduction of *UCTP*. Real world instances of university course timetabling are usually considered as large scale and NP-hard problems. Some exact algorithms for solving integer programming problems (such as the cutting planes and branch and cut) in the early development were not able to solve large scale problems. Therefore, many researchers apply heuristic or metaheuristic algorithms such as *Tabu* search, Genetic algorithm to solve this problem. Recent significant advances in both software and hardware computer technologies, make it possible for us to apply exact algorithms (such as branch and cut) for solving relatively large scale problems. However, there are many large scale problems that can not be solved by using exact algorithms yet. If it is possible to transform the original large scale

problem into some smaller problems, then we may be able to apply these exact methods and solve the problem efficiently.

In the following, if the scheduling of a part of the *UCTP* do not influence on other parts of the *UCTP*, then we call it an independent part. The basic idea of our presented approach is to first remove the rooms from the problem and identify independent parts of the *UCTP*. Then, we associate a priority for each independent part of the *UCTP* and consider the part with highest priority. In the next step, we heuristically estimate the number of required rooms for the selected part of *UCTP*. For example, if we have three courses and two teachers in the selected independent part, then we may heuristically estimate that two rooms are enough for scheduling these courses and teachers. Then, we apply the branch and cut algorithm to the scheduling problem consisting of the selected part and selected rooms and find its optimal solution. Next, we remove this part from the *UCTP* and consider the next independent part with highest priority. At this stage, there may exist some time periods of the selected rooms of the previous independent part, which have already scheduled for. We remove these time periods from the problem and in the remainder of the problem we assume that the corresponding rooms are not available on these time periods. We then estimate the required number of rooms for the new selected part and proceed as before. To identify the independent parts of the *UCTP*, we associate a graph to the *UCTP* as follows. Each node of this graph corresponds to a teacher of the university. If two teachers, teach some courses of the same group, then we define an edge between the corresponding nodes. Clearly, if we do not consider the rooms of the problem, then every component of this graph corresponds to an independent part of the *UCTP*.

Outline of the graph based reduction algorithm

Step 1: Define a node for each teacher.

Step 2: If two teachers teach some courses of the same group, then define an edge between the corresponding nodes.

Step 3: Find the components of the resulting graph.

Step 4: Assign a priority to each component of the graph.

Step 5: If the set of the components of the graph is empty, then stop. Otherwise, select the component of the graph with highest priority.

Step 6: Estimate the required number of rooms for scheduling the courses of the selected part and assign the estimated number of rooms to the selected part.

Step 7: Apply the branch and cut algorithm to the selected part and selected rooms to find the optimal solution.

Step 8: Assume that the selected rooms of the previous parts, are not available on the time periods that have already been scheduled for and go to Step 5.

In the following we characterize some features of our presented graph based reduction approach. The first feature of our presented approach is that it is possible to apply some traditional conventions and organizational rules to obtain smaller problems. For example, in many universities courses are usually scheduled for either even days (Saturday, Monday and Wednesday) or odd days (Sunday and Tuesday). In this case we can provide a list for courses requesting to be scheduled for even days and another list for courses requesting to be scheduled for odd days, and treat each list separately. The second feature of our presented approach is that it enables the managers of the university to know how they assign courses to teachers, so that the number of components of the associated graph to the *UCTP* is as large as possible and the resulting problems are smaller. For example, managers can assign a certain number of teachers to support the courses of each group. Some courses are common between students of several fields, the mangers can assign optimal time periods and rooms to these courses and remove the associated rooms, time periods and courses from the problem, to increase the number of components. One of the important parts of our proposed approach is the estimation of the required number of rooms for scheduling the courses of the selected part. The smaller the number of rooms, the faster the branch and cut algorithm. The minimal number of rooms can be easily determined by trial and error. For example, we can start by one room. If the branch and cut algorithm, informs us that the problem is infeasible, then we can try two rooms and proceed. In our presented approach, managers can determine the teachers that present common courses between several components of the graph and assign optimal times and rooms to them. This will increase the number of components and reduce the size of larger components of the graph.

٤. Numerical Results

In this section we study the numerical results. We used the Shiraz University of Technology (*Sutech*) course timetabling problem to examine the numerical efficiency of our proposed graph based reduction approach. At first we noted that in *Sutech* five days of the week are considered as the time horizon. These, five days are Saturday, Sunday, Monday, Tuesday and Wednesday. In the following, we call Saturday, Monday, Wednesday, even days and Sunday, Tuesday odd days. For each day we considered two sessions, namely morning and evening. In *Sutech* courses that should be scheduled for more than one working days, are usually scheduled for even days or odd days. For example, if a course is scheduled for Saturday evening, then its next session (if there exists any) is scheduled for Monday evening or Wednesday evening. Similarly, if a course is scheduled for Sunday morning, then its next session (if there exists any) is scheduled for Tuesday morning. For most courses, morning sessions of each working day have more desirability than evening sessions. Therefore, we will deal with the mornings of even days, mornings of odd days, evenings of even days and evenings of odd days separately, and call each of them a partition of the time horizon. Indeed, we first considered the morning of even days and provide the list of courses for which there is a request for scheduling in this partition. Then, we tried to schedule courses of the list in this partition. Then, we considered the morning of even days, the evening of even days and the evening of odd days, respectively, and repeated this procedure for them. Here, note that to examine the numerical efficiency of our proposed approach, it is enough to consider the largest partition of the time horizon (morning of even days). The timetabling of other partitions can be performed similarly.

Next, we study the properties of the partition of the time horizon corresponding to the morning of even days. In *Sutech*, morning session usually starts at 7:30 am and ends at 12:30 am. We assign to every thirty minutes a time period. Therefore, 7:30 am to 8:00 am is the first time period, 8:00 am to 8:30 am is the second time period and so on. This way, time periods number 1 to 10 determine Saturday morning, time periods number 11 to 20 determine Monday morning and time periods number 21 to 30 determine Wednesday morning. For example, if a course is scheduled for time periods number 12, 13 and 14, this means the course is scheduled for Monday morning 8:00 am to 9:30 am. Note that for morning of even days we have 30 time periods. Similarly, for the partition of the time horizon corresponding to morning of odd days we have 20 time periods and time periods 1 to 10 determine the Sunday morning and time periods 11 to 20 determine Tuesday morning.

In the next step of our proposed approach, we select a partition of the time horizon and provide a list of courses for scheduling in this partition. Each course is added to the list, if the teacher of the course selects this partition for teaching. If it is not possible to schedule all courses of the list in this partition, then we delete some courses with lower priority. A deleted course will be considered for scheduling in the partition corresponding to the next priority of the teacher of the course. In our proposed approach we give the highest priority to the *Phd* courses. The next priorities belong to the *MSc* and *BSc* courses, respectively. Among *BSc* courses, we give the highest priority to the courses of the last year and the next priorities belong to the course of the third year, second year and the first year, respectively. Here, we note that, courses may belong to different fields. We define priorities according to the population of fields. That means if field *A* has more population that field *B*, then it has higher priority. Therefore, *MSc* courses of field *A* has higher priority than *MSc* courses of field *B*. However, note that *MSc* courses of field *B* has higher priority than *BSc* courses of field *A*.

As we noted earlier, a group is a set of students attending the same courses. There are two semesters in every year. In each semester, students who are in the first year usually take the same courses, students who are in the second year usually take the same courses and so on. Therefore, in each semester, for each *BSc* field we have four groups, for each *MSc* field we have two groups and for each *Phd* field we have two groups. Clearly, it may not possible, to schedule all courses of a group, in one partition of the time horizon. Therefore, we can consider at most three courses of a group for scheduling for the morning partitions and at most two courses of a group for scheduling for the evening partitions.

In university course timetabling, some courses may be common between several groups. For example, in the first semester of the year, *BSc* students of the Electrical engineering, Information Technology and Industrial engineering who are in the first year of their education must take the course Calculus I. In these situations, we have to provide more than one section of Calculus I for them. These different sections of the same course, may or may not be taught by the same teacher. However, it is usually known that how many sections of the course should be taught by which teacher. Therefore, in our presented approach, we deal with different sections of a course as different courses. For example, if we have two sections of Calculus I, then we assume that we have two courses namely, Calculus I-section I and Calculus I-section II. This way, without loss of generality we can assume that each course is taught by exactly one teacher. Since, it may not be possible to schedule all courses of a teacher, in one partition, we heuristically consider at most three

courses of a teacher for scheduling for the morning partitions and at most two courses of a teacher for scheduling for the evening partitions.

After providing a list of courses for the selected partition of the time horizon, we associated a graph to the problem as described before. As we noted earlier, the key idea to our approach, is that if we associate different sets of rooms to the components of the graph, then the timetablings of different components are independent from each other. On the other hand, if we solve the timetablings of different components separately, then there might be some time periods that rooms associated to a component are free while for other components we conclude that the rooms are not enough for scheduling all courses. Another drawback with this strategy is that we may use a large number of rooms for timetabling, while the timetabling can be performed by using smaller number of rooms. Therefore, we first consider the smallest component of the graph and solve the timetabling problem for this component. Then, we consider the next smallest component, add some constraints to remove the time periods and rooms associated to the timetabling of the previous component and solve the timetabling problem for the current component. The timetabling of this partition of the time horizon, can be performed by repeating this procedure until all components of the graph are considered. Here, note that to examine the numerical efficiency of our proposed approach it is enough to consider the largest component of the graph associated to the morning of even days. Other, components can be dealt with similarly.

We considered 4 integer programming formulation for solving the timetabling problem, corresponding to the largest component. The first formulation consists of the objective function (9) and constraints (1-8) and is denoted *UCTP*. The second one consists of the objective function (9) and constraints (1-8) and valid inequalities (14-20) and is denoted *UCTPV*. The third one consists of the objective function (13) and constraints (1-8) and is denoted by *UCTPS*. The fourth one consists of the objective function (13) and constraints (1-8) and valid inequalities (14-20) and is denoted by *UCTPSV*. We compared these formulations according to the objective function value (*opt*), computing time (in seconds) (*time*), number of constraints (*cons*), number of variables (*vars*), number of nonzero entries (*nnz*), number of nodes processed by the branch and cut method (*nodes*), the required time to reach the first integer feasible solution (*ffs*) and the number of iterations (*iter*). The numerical results are recorded in Table 1. For the implementation we used the *CPLEX* software installed on Windows 10 operating system using a corei5 *CPU* with *2.5GHz* speed and 8G of *RAM*.

We first tried to solve the whole problem by using the *CPLEX* [22] but since the problem was large scale, we encountered the out of memory error. Then, we used our proposed graph based reduction to the problem as described before and the *CPLEX* becomes able to solve the largest component of the largest partition of the time horizon. Therefore, we believe that our proposed approach enables us to provide a (hopefully high quality) feasible solution to real world large scale university course timetabling problems.

The numerical results of Table 1 shows that *UCTP* have the same objective function value as *UCTPV* and *UCTPS* have the same objective function value as *UCTPVS*. Note that we cannot compare the objective function of *UCTP* and *UCTPV* with the objective function of *UCTPS* and *UCTPVS*. Moreover, adding valid inequalities to the formulation does not have any effect to the objective function value. However, we observed that the solutions obtained by the *UCTPS* and

UCTPSV are more desirable that those of the *UCTP* and *UCTPV*. *UCTP* and *UCTPV* need less computing time than the *UCTPS* and *UCTPVS*. Adding valid inequalities to the *UCTP* increased the computing time, while adding valid inequalities to the *UCTPS* reduced the computing time. *UCTP* and *UCTPV* need less number of iterations than the *UCTPS* and *UCTPVS*. In both cases, adding valid inequalities reduced the number of iterations. *UCTP* and *UCTPV* need less number of nodes than the *UCTPS* and *UCTPVS*. In both cases, adding valid inequalities reduced the number of nodes. In both cases, adding valid inequalities increased the number of constraints and nonzero entries. In both cases, adding valid inequalities does not have any influence on the number of variables. In some cases, it is enough for us to have a feasible solution of the problem. In these cases, *UCTPS* needs least computing time. After *UCTPS*, *UCTP* needs less computing time than *UCTPV* and *UCTPSV*. Finally, between the models with valid inequalities *UCTPSV* need less computing time than the *UCTPV*. In both cases, adding valid inequalities increased the computing time for finding the first feasible solution.

Table 1. Time Periods of morning of even days

day	***7:30 8:00***	***8:00 8:30***	***8:30 9:00***	***9:00 9:30***	***9:30 10:00***	***10:00 10:30***	***10:30 11:00***	***11:00 11:30***	***11:30 12:00***	***12:00 12:30***
Sat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Mon	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Wed	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**

Table 2. Time Periods of evening of even days

day	***14:00 14:30***	***14:30 15:00***	***15:00 15:30***	***15:30 16:00***	***16:00 16:30***	***16:30 17:00***	***17:00 17:30***	***17:30 18:00***
Sat	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Mon	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Wed	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**

Table 3. Time Periods of morning of odd days

day	***7:30 8:00***	***8:00 8:30***	***8:30 9:00***	***9:00 9:30***	***9:30 10:00***	***10:00 10:30***	***10:30 11:00***	***11:00 11:30***	***11:30 12:00***	***12:00 12:30***
Sun	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Tue	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**

Table 4. Time Periods of evening of even days

day	***14:00 14:30***	***14:30 15:00***	***15:00 15:30***	***15:30 16:00***	***16:00 16:30***	***16:30 17:00***	***17:00 17:30***	***17:30 18:00***
Sun	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Tue	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Table 5. Number of courses, teachers and groups

Degree	*#Courses*	*#Teachers*	*#Groups*
BSc	101	69	16
MSc	48	57	27
PhD	19	24	13

Table 6. Comparison of four models on the largest component

model	*cons*	*vars*	*nodes*	*iter*	*nnz*	*opt*	*time*	*ffs*
UCTP	279191	6820	240	36688	821268	6	87	45
UCTPV	280587	6820	101	21771	1093836	6	378	260
UCTPS	279191	6862	87221	1604192	821268	246	1425	40
UCTPSV	280587	6862	3208	89441	1093836	246	456	210

Acknowledgments

The author would like to thank the council of Shiraz university of technology for supporting this work.

References

[1] Arbaoui, T., Boufflet, J.P. and Moukrim, A. (2019), Lower bounds and compact mathematical formulations for spacing soft constraints for university examination timetabling problems, *Comput. Oper. Res.*, 106, 133–142.

[2] Avella, P., Vasiliev, I. (2005), A Computational Study of a Cutting Plane Algorithm for University Course Timetabling, Journal of Scheduling, 8(6), 497-514.

[3] Bashab, A., Ibrahim, A.O., AbedElgabar, E.E., Ismail, M.A., Elsafi, A., Ahmed, A. and Abraham, A. (2020), A systematic mapping study on solving university timetabling problems using meta-heuristic algorithms, *Neural Comput. Appl.*, 32(23), 17397–17432.

[4] Bettinelli, A., Cacchiani, V., Roberti, R. and Toth, P. (2015), An overview of curriculum-based course timetabling, *Top*, 23(2), 313–349.

[5] Burke, E.K. and Petrovic, S. (2002), Recent research directions in automated timetabling, *Eur. J. Oper. Res.*, 140(2), 266–280.

[6] Cardonha, A.C., Bergman, D. and Day, R. (2022), Maximizing student opportunities for in-person classes under pandemic capacity reductions, *Decis. Support Syst.*, 154, 113697.

[7] Carter, M.W. (1986), A survey of practical applications of examination timetabling algorithm, Operations Research, 34(2) 193-202.

[8] Ceschia, S. Di Gaspero, L. and Schaerf, A. (2022), Educational Timetabling: Problems, Benchmarks, and State-of-the-Art Results, *doi:* 10.1016/j.ejor.2022.07.011.

[9] Chen, R.M. and Shih, H.F. (2013), Solving university course timetabling problems using constriction particle swarm optimization with local search, *Algorithms*, 6(2), 227–244.

[10] Chen, M.C., Goh, S.L., Sabar, N.R. and Kendall, G. (2021), A survey of university course timetabling problem: perspectives, trends and opportunities, *IEEE Access*, 9, 106515–106529.

[11] Colajanni, G. and Daniele, P. (2021), A new model for curriculum-based university course timetabling, *Optimization Letters*, 15(5), 1601–1616.

[12] Even S., Itai A. and Shamir A. (1975), On the complexity of time table and multi-commodity flow problems, *16th annual symposium on foundations of computer science (sfcs 1975)*, pp. 184–193.

[13]	Gotlieb, C.C. (1963), The construction of class-teacher timetables. In Even, S., Itai, A. and Shamir, A. (ED), *IFIP congress*, vol. 62, pp. 73–77.
[14]	Kaur, M. and Saini, S. (2021), A review of metaheuristic techniques for solving university course timetabling problem, *Adv. Inf. Commun. Technol. Comput.*, 19–25.
[15]	Komijan, A.R. and Koupaei, M.N. (2015), A mathematical model for university course scheduling: a case study, *Int. J. Tech. Res. Appl.*, 3(19), 20–25, 2015.
[16]	V. Kralev, V. and Kraleva, R. (2017), A local search algorithm based on chromatic classes for university course timetabling problem, *Int. J. Adv. Comput. Res.*, 7(28), 1-7.
[17]	Murray, K., Muller, T. and Rudova, H. (2007), Modeling and solution of a complex university course timetabling problem. In Burke, E. and Rudova, H. (ED), Practice and Theory of Automated Timetabling, volume 3867 of Lecture Notes in Computer Science, Springer Heidelberg: Berlin, pp. 189-209.
[18]	Qualizza, A. and Paolo, S. (2005), A column generation scheme for faculty timetabling. In Burke, E. and Trick, M. (ED), Practice and Theory of Automated Timetabling, volume 3616 of Lecture Notes in Computer Science, Springer Heidelberg: Berlin, pp. 161-173.
[19]	Schaerf, A. (1999), A survey of automated timetabling, Artificial Intelligence Review, 13, 87-127.
[20]	de Werra, D. (1985), An introducing to timetabling, European Journal of Operational Research, 19,151-162.